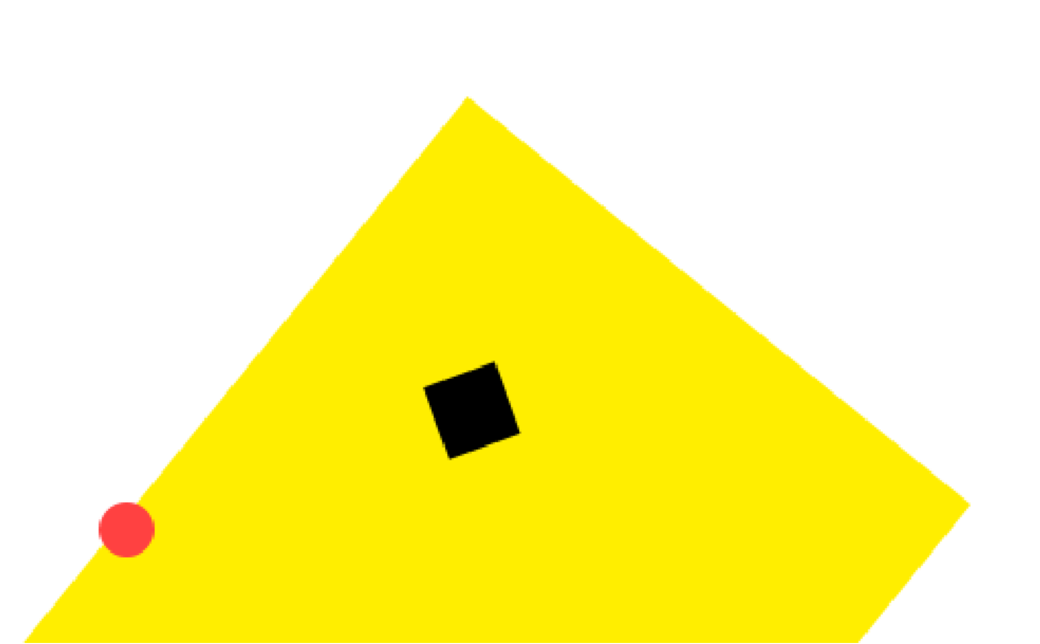
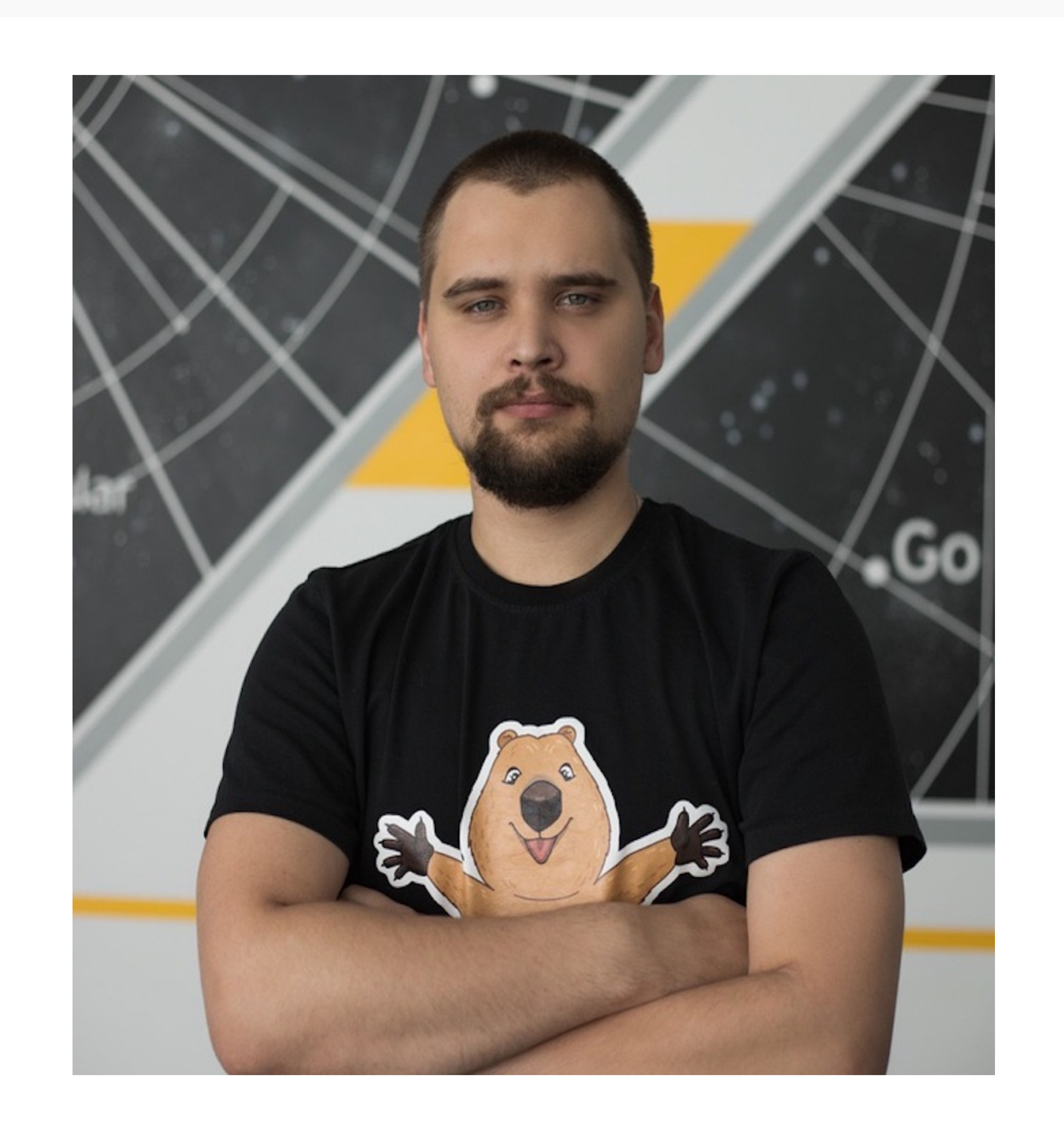
История, как мы на Module Federation съезжали

Смирнов Максим





Немного о себе



- 6 лет занимаюсь frontend-разработкой
- 4 года являюсь частью Тинькофф
- Начал разработчиком и дорос до руководителя направления
- Во время работы был как фронтом, так и бэкером с уклоном в SRE





Вводная

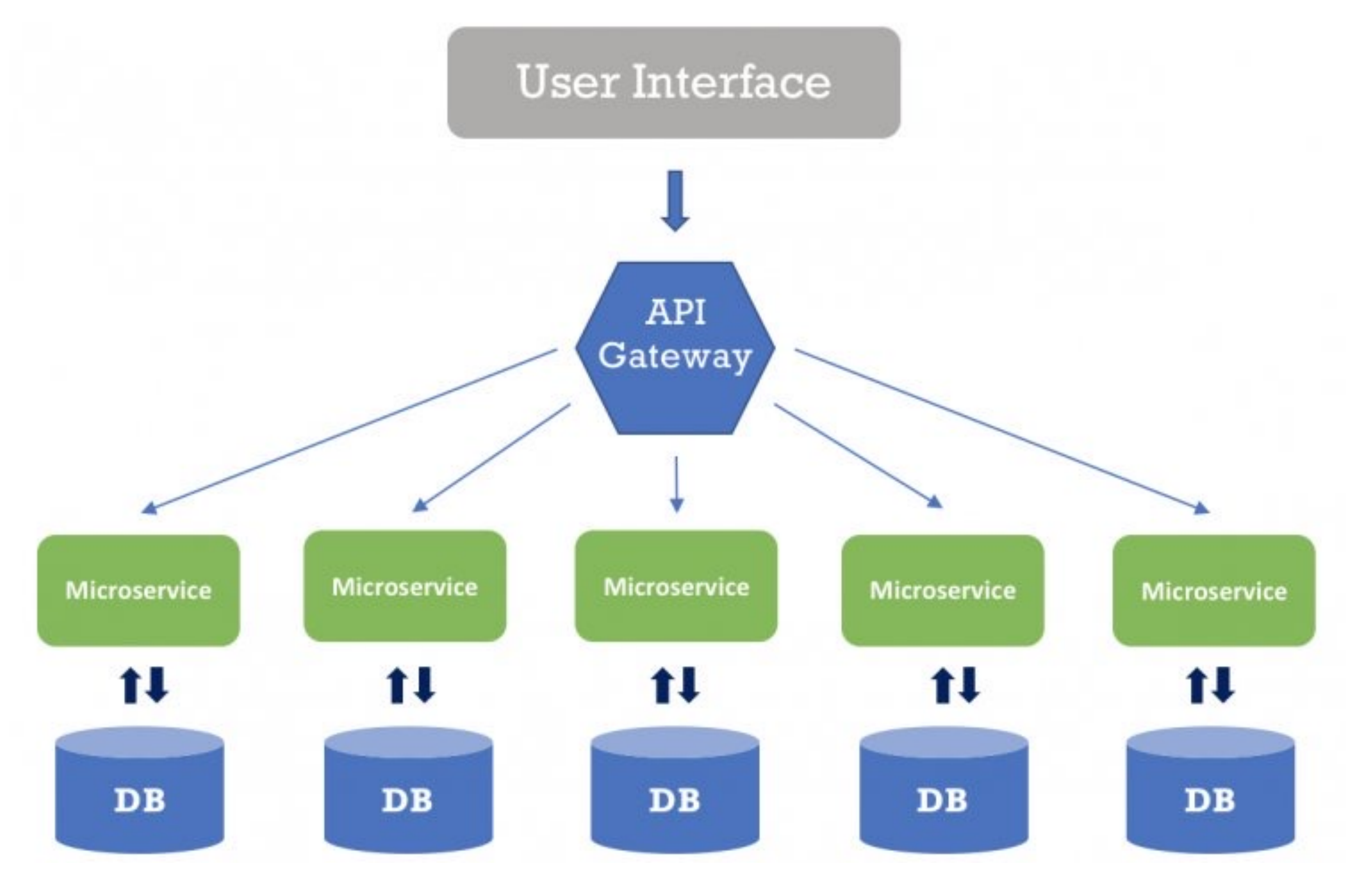
- Монолит – сборная солянка из кода с примесью спагетти





Вводная

Микросервис – сервис исполняющий только определенную задачу



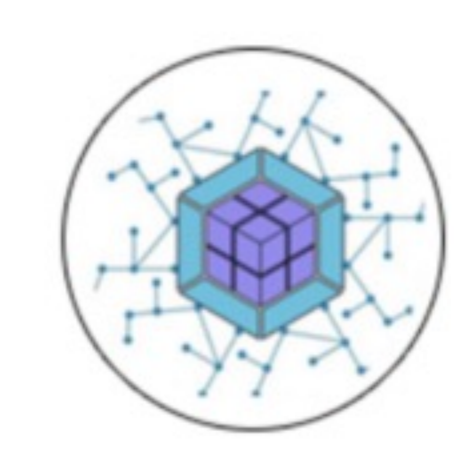
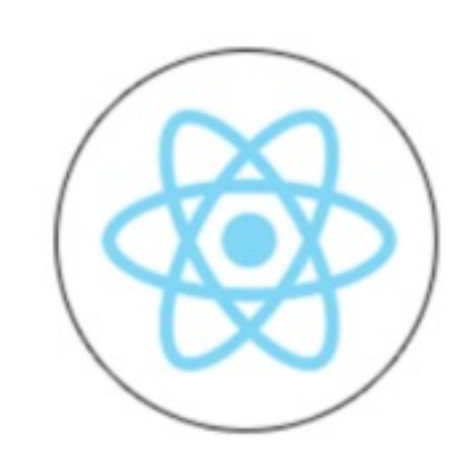




Вводная

Module Federation (MFE) – Webpack-плагин для управления микросервисами







О чем будем говорить?

- Каков был тот монолит?





О чем будем говорить?

- Каков был тот монолит?
- Как мы приняли решение пилиться?





О чем будем говорить?

- Каков был тот монолит?
- Как мы приняли решение пилиться?
- Что мы накрутили в MFE





О чем будем говорить?

- Каков был тот монолит?
- Как мы приняли решение пилиться?
- Что мы накрутили в MFE
- Бонусы, полученные от микрофронтендового подхода





О чем будем говорить?

- Каков был тот монолит?
- Как мы приняли решение пилиться?
- Что мы накрутили в MFE
- Бонусы, полученные от микрофронтендового подхода
- Куда дальше будем его развивать





Тот самый монолит

Classic personal account

Routing

Components

Business logic

API Layer

Backend API





Тот самый монолит

Root /

/main

/operations

/stores

/settings

/operations/list

/operations/info/:id

/operations/info/:id/general

/operations/info/:id/items





Что мы решили делать? #1

- Перевод проекта на монорепозиторий nx-workspace





Что мы решили делать? #1

- Перевод проекта на монорепозиторий nx-workspace
- Вынести в лейзи-модули все, что не было вынесено раньше





Что мы решили делать? #1

- Перевод проекта на монорепозиторий nx-workspace
- Вынести в лейзи-модули все, что не было вынесено раньше
- Определить границы роутинга внутри страниц





Что мы решили делать? #1

- Перевод проекта на монорепозиторий nx-workspace
- Вынести в лейзи-модули все, что не было вынесено раньше
- Определить границы роутинга внутри страниц
- Декомпозиция логики по локальным/npm-библиотекам





Что мы решили делать? #1

- Перевод проекта на монорепозиторий nx-workspace
- Вынести в лейзи-модули все, что не было вынесено раньше
- Определить границы роутинга внутри страниц
- Декомпозиция логики по локальным/npm-библиотекам
- Шина данных между страницами





Что мы решили делать? #1

- Перевод проекта на монорепозиторий nx-workspace
- Вынести в лейзи-модули все, что не было вынесено раньше
- Определить границы роутинга внутри страниц
- Декомпозиция логики по локальным/npm-библиотекам
- Шина данных между страницами
- Перевод больших страниц на фича-сторы





Что вышло? #1

- Чуть более красивый монолит
- Вся общая логика находится в библиотеках
- Единый подход по получения глобальных данных
- Все наконец-то переписано на лейзи-модули





SHOULD I DEPLOY TODAY?

I SEE YOU DEPLOYED ON FRIDAY

Сбой как мотиватор распила

Сбой в монолите





Сбой как мотиватор распила

Сбой в монолите

- Недоступность всего функционала





Сбой как мотиватор распила

Сбой в монолите

- Недоступность всего функционала
- Сложная идентификация триггера сбоя





Сбой как мотиватор распила

Сбой в монолите

- Недоступность всего функционала
- Сложная идентификация триггера сбоя
- Постоянные подгоны со стороны бизнеса





Сбой как мотиватор распила

Сбой в монолите

- Недоступность всего функционала
- Сложная идентификация триггера сбоя
- Постоянные подгоны со стороны бизнеса
- Оптимизации после сбоев как триггер для нового сбоя





Сбой как мотиватор распила

Сбой в микросервисе





Сбой как мотиватор распила

Сбой в микросервисе

- Недоступность только части функционала





Сбой как мотиватор распила

Сбой в микросервисе

- Недоступность только части функционала
- Легкий поиск триггера





Сбой как мотиватор распила

Сбой в микросервисе

- Недоступность только части функционала
- Легкий поиск триггера
- Подгоны от бизнеса будут всегда





Сбой как мотиватор распила

Сбой в микросервисе

- Недоступность только части функционала
- Легкий поиск триггера
- Подгоны от бизнеса будут всегда
- Полечить место сбоя легче, все замкнуто на микросервис





Что мы придумали

Host App

settings page

main app

operations app

stores app

NestJS BFF

backend microservice 1

backend microservice 4

backend microservice 3

backend microservice 2

Public network zone

Private network zone

Что мы придумали

Header
Main
Stores
Operations
USR

Remote app content

Footer





MFE, я выбираю тебя





Как оно выглядит вначале

```
plugins: [
    new ModuleFederationPlugin({
        remotes: {
            main: 'main@http://localhost:4201/remoteEntry.js',
            operations: 'operations@http://localhost:4202/remoteEntry.js',
            stores: 'stores@http://localhost:4203/remoteEntry.js',
        },
        shared: share({
            '@angular/core': {singleton: true, strictVersion: true, requiredVersion: 'auto'},
            '@angular/common': {singleton: true, strictVersion: true, requiredVersion: 'auto'},
            '@angular/common/http': {singleton: true, strictVersion: true, requiredVersion: 'auto'},
            '@angular/router': {singleton: true, strictVersion: true, requiredVersion: 'auto'},
            '@angular/forms': {singleton: true, strictVersion: true, requiredVersion: 'auto’},
           ...sharedMappings.getDescriptors(),
        }),
    }),
    sharedMappings.getPlugin(),
],
```

FC Frontend Conf 2022

Как оно выглядит вначале

```
remotes: {
   main: 'main@http://localhost:4201/remoteEntry.js',
   operations: 'operations@http://localhost:4202/remoteEntry.js',
   stores: 'stores@http://localhost:4203/remoteEntry.js',
},
```



Как оно выглядит вначале

```
shared: share({
    '@angular/core': {singleton: true, strictVersion: true, requiredVersion: 'auto'},
    '@angular/common': {singleton: true, strictVersion: true, requiredVersion: 'auto'},
    '@angular/common/http': {singleton: true, strictVersion: true, requiredVersion:
    'auto'},
    '@angular/router': {singleton: true, strictVersion: true, requiredVersion: 'auto'},
    '@angular/forms': {singleton: true, strictVersion: true, requiredVersion: 'auto’},
    ...sharedMappings.getDescriptors(),
}),
```



Как оно выглядит вначале

Плюсы





Как оно выглядит вначале

Плюсы

- Работает из коробки





Как оно выглядит вначале

Плюсы

- Работает из коробки
- Удобное управление зависимостями





Как оно выглядит вначале

Плюсы

- Работает из коробки
- Удобное управление зависимостями
- Из конфига видно сразу все приложения





Как оно выглядит вначале

Вопрос-наброс





Как оно выглядит вначале

Вопрос-наброс

- А как же управлять динамически?





Как оно выглядит вначале

Вопрос-наброс

- А как же управлять динамически?
- Что там по фолбэкам?





Как оно выглядит вначале

Вопрос-наброс

- А как же управлять динамически?
- Что там по фолбэкам?
- А что, если я хочу ремоут в ремоуте?





Что там по динамике

- Взять файл с конфигом





Что там по динамике

- Взять файл с конфигом
- Перетряхнуть роутер





Что там по динамике

- Взять файл с конфигом
- Перетряхнуть роутер
- Добавить Lazy Loading для remote-модулей





Что там по динамике

HOST

1. Get config

/assets/config.json

2. Build router tree

/main

/operations

/stores

3. Lazy load

mainApp/remoteEntry.js





Что там по динамике

```
[
    {
        "remoteEntry": "http://localhost:4201/remoteEntry.js",
        "remoteName": "main",
        "exposedModule": "./Module",
        "displayName": "navigation.main",
        "routePath": "main",
        "ngModuleName": "RemoteEntryModule"
    },
    {
        "remoteEntry": "http://localhost:4202/remoteEntry.js",
        "remoteName": "stores",
        "exposedModule": "./Module",
        "displayName": "navigation.stores",
        "routePath": "stores",
        "ngModuleName": "RemoteEntryModule"
    }
]
```




Загрузчик

```
import {LoadRemoteModuleOptions} from '@angular-architects/module-federation';

export type Microfrontend = LoadRemoteModuleOptions & {
    remoteName: string;
    displayName: string;
    routePath: string;
    ngModuleName: string;
};
```





Загрузчик

```
import {LoadRemoteModuleOptions} from '@angular-architects/module-federation';

export type Microfrontend = LoadRemoteModuleOptions & {
    remoteName: string;
    displayName: string;
    routePath: string;
    ngModuleName: string;
};
```





Загрузчик

```
export class MicrofrontendLoaderService {
	constructor(
		private readonly router: Router,
		private readonly httpClient: HttpClient,
		private readonly destroy$: TuiDestroyService
	) {}


buildDynamicRoutes(): Observable<boolean> {
	return this.resolveConfig().pipe(
		takeUntil(this.destroy$),
		tap(cfg =>
			this.router.resetConfig(
				buildApplicationRoutes(cfg),
			),
		),
		mapTo(true),
	)
}

private resolveConfig(): Observable<Microfrontend[]> {
   return this.httpClient.get<Microfrontend[]>('/assets/config/mf/config.json')
}
```



Загрузчик

```
private resolveConfig(): Observable<Microfrontend[]> {
   return this.httpClient.get<Microfrontend[]>('/assets/config/mf/config.json')
}
```



Загрузчик

```
buildDynamicRoutes(): Observable<boolean> {
    return this.resolveConfig().pipe(
        takeUntil(this.destroy$),
        tap(cfg =>
            this.router.resetConfig(
                buildApplicationRoutes(cfg),
             ),
        ),
        mapTo(true),
    )
}
```



Загрузчик

```
providers: [
    {
        provide: APP_INITIALIZER,
        useFactory: (microfrontendService: MicrofrontendService) => () =>
        microfrontendService.buildDynamicRoutes(),
        deps: [MicrofrontendService],
        multi: true,
    },
],
```





Роутер-шейкер

```
import {loadRemoteModule} from '@angular-architects/module-federation';

export function buildApplicationRoutes(options: Microfrontend[]): Routes {
	const mfRoutes: Routes = Array.from(options).map(o => ({
		path: o.routePath,
		loadChildren: () => loadRemoteModule(o).then(m => m[o.ngModuleName]),
		canActivate: [AuthGuard],
	}));

	return [
		...mfRoutes,
		{
			path: '',
			redirectTo: 'main',
			pathMatch: 'full',
		},
		{
			path: '**',
			redirectTo: '404',
			pathMatch: 'full',
		},
	];
}
```



Роутер-шейкер

```
import {loadRemoteModule} from '@angular-architects/module-federation';

export function buildApplicationRoutes(options: Microfrontend[]): Routes {
    const mfRoutes: Routes = Array.from(options).map(o => ({
        path: o.routePath,
        loadChildren: () => loadRemoteModule(o).then(m => m[o.ngModuleName]),
        canActivate: [AuthGuard],
    }));

    return [
        ...mfRoutes,
        {
            path: '',
            redirectTo: 'main',
            pathMatch: 'full',
        },
        {
            path: '**',
            redirectTo: '404',
            pathMatch: 'full',
        },
    ];
}
```



Роутер-шейкер

```
const mfRoutes: Routes = Array.from(options).map(o => ({
   path: o.routePath,
   loadChildren: () => loadRemoteModule(o).then(m => m[o.ngModuleName]),
   canActivate: [AuthGuard],
}));
```



Что там по динамике







Динамические загрузки – плюшки

- Кэширование конфига в коде при обращении





Динамические загрузки – плюшки

- Кэширование конфига в коде при обращении
- Подкладывание конфига в session storage





Динамические загрузки – плюшки

- Кэширование конфига в коде при обращении
- Подкладывание конфига в session storage
- Конфиг как отдельная репа с CI/CD и безрелизное добавление новых сервисов





Динамические загрузки — плюшки

- Кэширование конфига в коде при обращении
- Подкладывание конфига в session storage
- Конфиг как отдельная репа с CI/CD и безрелизное добавление новых сервисов
- CDN для раздачи конфига





Фолбэки









Фолбэки

- Загрузка конфига





Фолбэки

- Загрузка конфига
- Переходы по приложению





Фолбэки

- Загрузка конфига
- Переходы по приложению
- Плановые работы на remote-приложении





Фолбэк: Упавший конфиг

HOST

Запрос конфига

NGINX

config.json

Запрос конфига

Да

Нет

Есть в кэше?

Сгоняй за файлом

Нет

Да

Кэш валиден?

Сервис живой?

Да

OPTIONS запрос к серверу с конфигом

Забери файл и положи в кэш

Нет

Отдай клиенту

Отдаем клиенту файл из протухшего кэша





Фолбэк: Сломанный переход

Как говорится, просто добавь ~~воды~~ обработку и редирект в host-приложении

```
this.router.events.pipe(takeUntil(this.destroy$)).subscribe((event: Event) => {
  switch (true) {
    case event instanceof NavigationError: {
      /*
        Do some logic
        .....
      */
      this.router.navigate(['redirect', 'error']);

      break;
    }
  }
});
```





Фолбэк: Работы на проде





Фолбэк: Работы на проде

```
{
   "remoteEntry": "http://localhost:4201/remoteEntry.js",
   "remoteName": "main",
   "exposedModule": "./Module",
   "displayName": "navigation.main",
   "routePath": "main",
   "ngModuleName": "RemoteEntryModule"
   "health": {
      "url": "/remote/main/health",
      "ttl": "5000",
      "scheduler": "10"
   }
},
```



ТИНЬКОФФ



Фолбэк: Работы на проде

HOST

Запрашиваем хелс

Ответ получен?

Да

RC 200?

Да

Все хорошо, живем как раньше

Нет

Нет

Создай алерт, залогируй, повесь плашку в хедере о временной недоступности функциональности





Ремоут в ремоуте

НУ ВЫ ПОНЯЛИ...



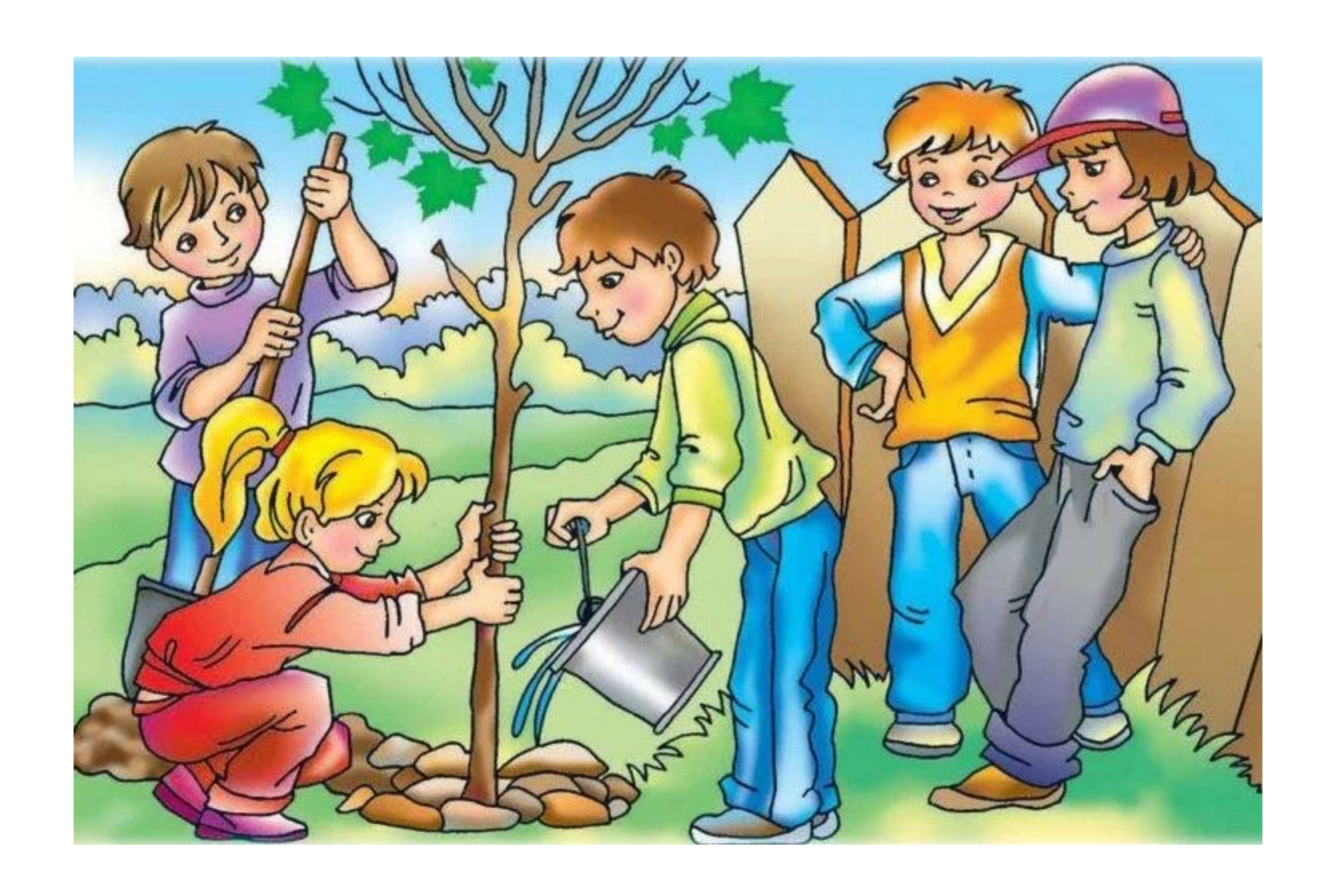


Небольшой рассказ





Небольшой рассказ







Небольшой рассказ





Ремоут в ремоуте

- Есть два проекта: ЛК и Админка





Ремоут в ремоуте

- Есть два проекта: ЛК и Админка
- Админы хотят видеть определенную страницу ЛК у себя





Ремоут в ремоуте

- Есть два проекта: ЛК и Админка
- Админы хотят видеть определенную страницу ЛК у себя
- Не иметь часть функциональности





Ремоут в ремоуте

- Есть два проекта: ЛК и Админка
- Админы хотят видеть определенную страницу ЛК у себя
- Не иметь часть функциональности
- А главное: быстро





Ремоут в ремоуте

Header

APPS

APPS

APPS

APPS

Remote content

Sub route

Sub route

Sub route

Remote route

Footer

ADM





Ремоут в ремоуте

ХОСТ В КОТОРОМ РЕМОУТ

ВНУТРИ КОТОРОГО ЕЩЕ ОДИН РЕМОУТ

risovach.ru





Кодируем на успех

```
private resolveConfig(): Observable<Microfrontend[]> {
  return this.httpClient.get<Microfrontend[]>('/assets/config/mf/config.json').pipe(
    tap(mfConfig =>
        sessionStorage.setItem(CONFIGURATION_STORAGE_KEY, JSON.stringify(mfConfig)))
    )
}
```





Кодируем на успех

```
export function loadRemoteRoute(remoteName: RemoteNames) {
  return new Promise((resolve, reject) => {
    const mfConfig = sessionStorage.getItem(CONFIGURATION_STORAGE_KEY);

    if (!mfConfig) {
      reject(new Error('CONFIG not provided'));
    }

    const remotes: Microfrontend[] = JSON.parse(mfConfig as string);
    const requiredRemote = remotes.find((remote: Microfrontend) => remote.remoteName ===
    remoteName);

    if (!requiredRemote) {
      reject(new Error(`Remote with provided name ${remoteName}, not found`));
    }

    resolve(
      loadRemoteModule(requiredRemote as Microfrontend).then(m => m[(requiredRemote as
    Microfrontend).ngModuleName]),
    );
  });
}
```

Кодируем на успех

```
export function loadRemoteRoute(remoteName: RemoteNames) {
  return new Promise((resolve, reject) => {
    const mfConfig = sessionStorage.getItem(CONFIGURATION_STORAGE_KEY);

    if (!mfConfig) {
      reject(new Error('CONFIG not provided'));
    }

    const remotes: Microfrontend[] = JSON.parse(mfConfig as string);
    const requiredRemote = remotes.find((remote: Microfrontend) => remote.remoteName ===
    remoteName);

    if (!requiredRemote) {
      reject(new Error(`Remote with provided name ${remoteName}, not found`));
    }

    resolve(
      loadRemoteModule(requiredRemote as Microfrontend).then(m => m[(requiredRemote as
    Microfrontend).ngModuleName]),
    );
  });
}
```

Кодируем на успех

```
export function loadRemoteRoute(remoteName: RemoteNames) {
  return new Promise((resolve, reject) => {
    const mfConfig = sessionStorage.getItem(CONFIGURATION_STORAGE_KEY);

    if (!mfConfig) {
      reject(new Error('CONFIG not provided'));
    }

    const remotes: Microfrontend[] = JSON.parse(mfConfig as string);
    const requiredRemote = remotes.find((remote: Microfrontend) => remote.remoteName ===
    remoteName);

    if (!requiredRemote) {
      reject(new Error(`Remote with provided name ${remoteName}, not found`));
    }

    resolve(
      loadRemoteModule(requiredRemote as Microfrontend).then(m => m[(requiredRemote as
    Microfrontend).ngModuleName]),
    );
  });
}
```

ТИНЬКОФФ



Кодируем на успех

```
export function loadRemoteRoute(remoteName: RemoteNames) {
  return new Promise((resolve, reject) => {
    const mfConfig = sessionStorage.getItem(CONFIGURATION_STORAGE_KEY);

    if (!mfConfig) {
      reject(new Error('CONFIG not provided'));
    }

    const remotes: Microfrontend[] = JSON.parse(mfConfig as string);
    const requiredRemote = remotes.find((remote: Microfrontend) => remote.remoteName ===
remoteName);

    if (!requiredRemote) {
      reject(new Error(`Remote with provided name ${remoteName}, not found`));
    }

    resolve(
      loadRemoteModule(requiredRemote as Microfrontend).then(m => m[(requiredRemote as
Microfrontend).ngModuleName]),
    );
  });
}
```

Кодируем на успех

```
{
  path: ’operations',
  loadChildren: () => loadRemoteRoute(‘operations'),
},
```



Шина данных – Event model

- Host-приложение выступает в роли Producer





Шина данных – Event model

- Host-приложение выступает в роли Producer
- Remote's выступают в роли consumer и ждут событий





Шина данных – Event model

- Host-приложение выступает в роли Producer
- Remote’s выступают в роли consumer и ждут событий

HOST

Producer

Set User

Language switch

....

REMOTE

Consumer

Подписывается на события и при их появление отрабатывает





Шина данных – State model

- Единый root-state для всех host-приложений





Шина данных – State model

- Единый root-state для всех host-приложений
- Единый набор селекторов/событий





Шина данных – State model

- Единый root-state для всех host-приложений
- Единый набор селекторов/событий
- Упаковка root-state в npm-либу





State model

HOST

STATE

ROOT STATE LIBRARY

REMOTE

1

1

1

get user info





Выводы

- Микрофронты -
    - универсальное средство решения проблем





Выводы

- Микрофронты -
    - универсальное средство решения проблем
    - это не про фреймворк, а про мышление





Выводы

- Микрофронты -
    - универсальное средство решения проблем
    - это не про фреймворк, а про мышление
- Любое коробочное решение требует допила под ваши задачи





Выводы

- Микрофронты -
    - универсальное средство решения проблем
    - это не про фреймворк, а про мышление
- Любое коробочное решение требует допила под ваши задачи
- В работе с любым приложением главное - фантазия





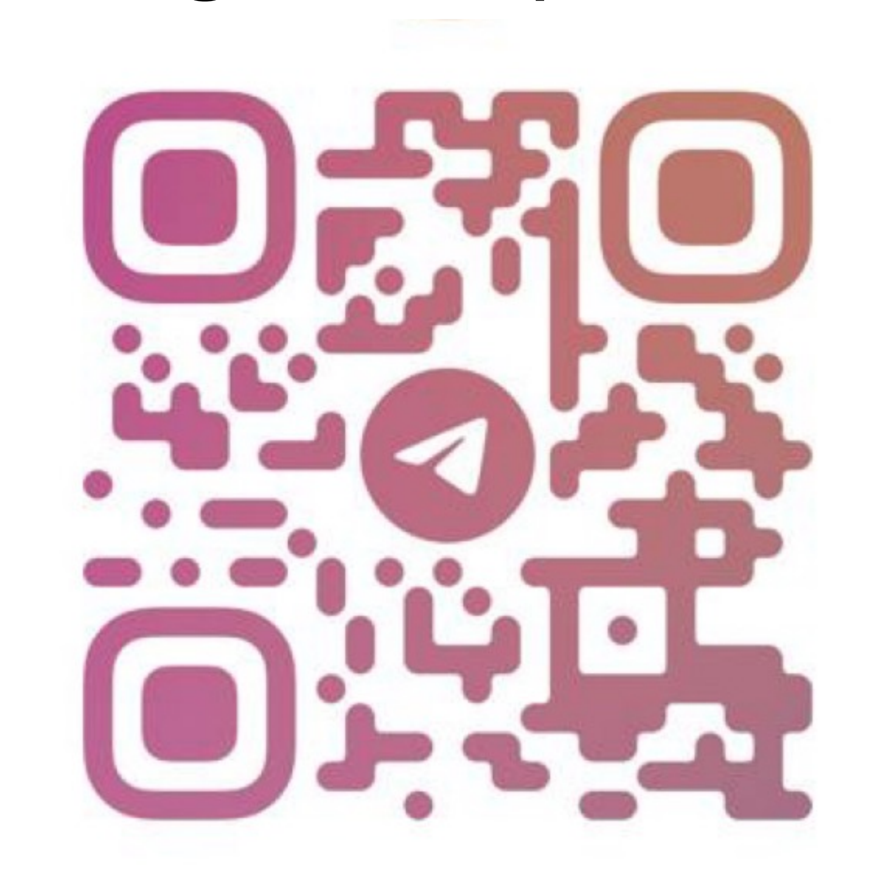
Выводы

- Микрофронты -
  - универсальное средство решения проблем
  - это не про фреймворк, а про мышление
- Любое коробочное решение требует допила под ваши задачи
- В работе с любым приложением главное - фантазия
- Экспериментируйте ☺





Смирнов Максим
Telegram: https://t.me/mvsmirn





Вопросы?



Голосование за доклад



